**WHAT IS CLAIMED IS:**

1. A computer architecture supporting interleaved execution of multiple threads, comprising:

     a processor adapted to initiate instructions associated with a thread;

     a commutator adapted to sequentially select threads for instruction initiation by the processor; and

     a cycle allocation table operably coupled to the commutator, comprising an execution time individually allotted to each of the threads.

2. The computer architecture as recited in claim 1, wherein said allotted execution times are not the same for all the threads.

3. The computer architecture as recited in claim 1, wherein the cycle allocation table is capable of being reconfigured during runtime, allowing the execution time allotted to the threads to be independently modified.

4. The computer architecture as recited in claim 1, wherein the configuration of the cycle allocation table and the execution time allotted to the threads is fixed and determined when the processor is designed or manufactured, and cannot be modified during runtime.

5. The computer architecture as recited in claim 1, wherein the commutator comprises a circular (i.e., modulo N) counter generating addresses in the cycle allocation table.

6. The computer architecture as recited in claim 1, further comprising a thread identifier and an execution context associated with each thread, wherein the execution context comprises a program counter and register set.

7. The computer architecture as recited in claim 1, wherein the processor execution time represents a number of clock cycles.

8. The computer architecture as recited in claim 1, wherein the cycle allocation table contains thread identifiers and the processor execution time allotted to a thread is based on the number of occurrences of its thread identifier in the cycle allocation table.

9. The computer architecture as recited in claim 1, wherein the processor execution time allocated to a given thread is commensurate with the workload of the thread.

10. The computer architecture as recited in claim 6, further comprising pipeline control and memory transaction logic, wherein said logic ensures that results of a pipeline operation or memory transaction are directed to the register set belonging to the thread that issued the operation or transaction.

11. The computer architecture as recited in claim 6, further comprising pipeline/register interlock logic adapted to prevent access to a register awaiting a result from a pending transaction until the result is returned.

12. The computer architecture as recited in claim 8, wherein the processor may initiate from 0 to N instructions each clock cycle, where $N \geq 1$.

13. The computer architecture as recited in claim 8, wherein the processor has no instruction pipeline, and initiates one instruction per clock cycle.

14.     The computer architecture as recited in claim 6, wherein there are 16 threads, wherein each register set comprises 32 registers, and wherein the cycle allocation table comprises 64 entries.

5     15.     A method for interleaved execution of a plurality of threads by a computer processor, comprising:

        assigning each thread a program counter, register set and thread identifier;

10      allotting a portion of an execution time of the processor to each thread, wherein a
            size of the portion may be different for each thread, and wherein the size
            of each portion comprises an allocation of the execution time; and

        executing simultaneously the plurality of threads, with each thread executing for
15          its allotted instruction initiation time.

16.     The method as recited in claim 15, wherein said allotting a portion of an execution time comprises placing one or more occurrences of the thread identifier in a cycle allocation table.

20

17.     The method as recited in claim 16, wherein threads are executed in the order in which their thread identifiers appear in the cycle allocation table.

18.     The method as recited in claim 16, wherein the portion of the execution time
25  allotted to a thread is based on the number of occurrences of its thread identifier in the cycle allocation table.

19.     The method as recited in claim 15, wherein the allotted execution time for a given thread is commensurate with the workload of the thread.

30

20.    The method as recited in claim 15, wherein the allotted execution time for a given thread is commensurate with the real time requirement of the program to which the thread belongs.

5    21.    The method as recited in claim 15, wherein the allotment of execution times to threads may be reapportioned during runtime.

22.    The method as recited in claim 15, wherein the result of an operation performed by the computer processor is directed to the register set belonging to the thread that issued 10    the operation or transaction, regardless of which thread is executing when the result becomes available.

23.    The method as recited in claim 15, wherein access to a register awaiting a result from a pending transaction is prevented until the result is returned.

15